НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський політехнічний інститут імені Ігоря Сікорського»

Затверджую
Голова приймальної комісії
Ректор

Михайло
ЗГУРОВСЬКИЙ

28.04.2023
дата

Навчально-науковий інститут прикладного системного аналізу
повна назва факультету/навчально-наукового інституту

ПРОГРАМА
комплексного фахового випробування
для вступу на освітньо-професійну програму підготовки магістра
«Інтелектуальні сервіс-орієнтовані розподілені обчислювання»

за спеціальністю 122 Комп'ютерні науки

Програму ухвалено:
Вченою Радою Навчально-наукового інституту
прикладного системного аналізу
Протокол № 3 від «27» березня 2023 р.
Заступник Голови Вченої Ради
_____ Віктор РОМАНЕНКО

Київ – 2023

ВСТУП

Програму комплексного фахового випробування для вступу на програму підготовки магістра за спеціальністю 122 Комп'ютерні науки (далі — Програма) призначено для отримання досвіду самостійної роботи абітурієнта з підготовки до екзамену. Програму вступного випробування розроблено атестаційною комісією навчально-наукового інституту прикладного системного аналізу, ухвалено Вченою радою НН ІПСА та затверджено Головою Приймальної комісії.

Метою програми є формування у абітурієнтів здатності ознайомитися із предметними питаннями курсів навчальних дисциплін, що включені в екзаменаційні білети; опрацювати підручники, навчальні посібники та інші інформаційно-літературні джерела предметної області знання; осмислено впорядкувати і систематизувати засвоєні теоретичні знання і практичні навички; мотивовано виконати роботу на екзамені, продемонструвавши певний рівень засвоєння навчальних дисциплін в результаті навчання.

Перелік навчальних дисциплін цієї Програми складають такі, що належать до циклу дисциплін базової підготовки навчального плану підготовки бакалавра напряму 122 Комп'ютерні науки:

1) Методи оптимізації;
2) Чисельні методи;
3) Об'єктно-орієнтоване програмування;
4) Операційні системи;
5) Крос-платформне програмування.

Вступний екзамен проводиться чотири академічних години без перерви (180 хвилин) у спосіб одержання екзаменаційного білету — повернення письмової роботи. Метою на екзамені є розв'язання завдань екзаменаційного білету. Екзаменаційний білет містить чотири практичних за типом завдання. Диференціації робочого часу, відведеного на виконання кожного завдання, немає. Фіксується час початку і закінчення роботи.

ПЕРЕЛІК НАВЧАЛЬНОГО МАТЕРІАЛУ, ЩО ВИНОСИТЬСЯ НА ФАХОВЕ ВИПРОБУВАННЯ

1. Навчальна дисципліна «Методи оптимізації»

Лінійне програмування (ЛП)

Метод розв'язання задач ЛП з довільним видом обмежень, оснований на штучних змінних. Двоїста задача ЛП. Двоїстий симплекс-метод. Метод оберненої матриці. Дослідження моделей ЛП-задач на чутливість. Транспортні задачі. Метод потенціалів.

Дискретне програмування (ДП)

Метод відсікаючих площин Гоморі. Метод гілок та меж. Метод гілок та меж в задачі комівояжера. Метод послідовного аналізу та відсіву варіантів (ПАВ) в

задачі ЛЦП. Метод послідовного аналізу та відсіву варіантів (ПАВ) в задачі булево програмування.

Нелінійне програмування

Методи пошуку одновимірного мінімуму (метод дихотомії, метод «золотого перетину», метод ДСК-Пауєла). Метод множників Лагранжа.

Задача квадратичного програмування. Умови оптимальності Куна-Таккера для задач квадратичного програмування. Базові методи безумовної мінімізації (градієнтні, квазіньютонівські).

Методи оптимізації з обмеженнями (методи «штрафних функцій», метод множників Лагранжа, проективні методи).

Методи можливих напрямків. Метод Зойтендейка у випадку лінійних обмежень. Метод Зойтендейка у випадку нелінійних обмежень-нерівностей.

Список літератури [1 — 3]

2. Навчальна дисципліна «Чисельні методи»

[Студенти демонструють знання чисельних методів розв’язку реальних задач, описуваних довільними нелінійними диференціально-алгебраїчними рівняннями великої розмірності. Освоївши такі методи, майбутній фахівець набуває здібностей до системного аналізу через математичне моделювання складних задач сучасної науки і техніки. Вивчення чисельних методів стимулює переосмислення і більш глибоке розуміння математики в цілому, оскільки алгоритми чисельних методів часто прямо ілюструють такі поняття, як збіжність, границя, нескінченно мала величина тощо. До програми вступного випробування включаються нижченаведені питання]

Похибки обчислень. Види похибок. Оцінювання похибки результату арифметичних операцій.

Розв’язання нелінійних рівнянь. Чисельні методи пошуку коренів рівняння: метод бісекції (половинного ділення), метод простої ітерації, метод січних, метод Ньютона, комбінований метод. Умови збіжності методів.

Прямі методи розв’язування систем лінійних алгебраїчних рівнянь. Метод Гаусса та його варіанти (*LU*, *LDU*, без зворотного ходу, матричний метод), метод квадратного кореня. Обчислення визначника системи, оберненої матриці. Умови збіжності методів. Обумовленість системи рівнянь.

Ітераційні методи розв’язування систем лінійних алгебраїчних рівнянь. Методи Якобі, Зайделя.

Наближення функцій. Задачі інтерполяції та апроксимації. Інтерполяційні формули Ньютона та Лагранжа. Інтерполяційний поліном Ерміта. Інтерполяція сплайнами. Метод найменших квадратів. Поточкова та інтегральна постановка. Оцінка похибок інтерполяційних формул.

Чисельне диференціювання. Оцінювання порядку точності різницевих формул.

Чисельне інтегрування. Формули середніх, трапецій, Сімпсона. Квадратурні формули Ейлера.

Спектральна задача. Методи: степеневий, скалярних добутків, Данилевського, Крилова, Якобі, QR, LR, обернених ітерацій. Перетворення подібності, конґруентне, Ґаусголдера.

Розв’язування задачі Коші для звичайних диференціальних рівнянь. Одно- та багатокрокові методи. Методи Ейлера, Рунге-Кутти першого, другого, четвертого порядків. Явні та неявні методи Адамса першого, другого, четвертого порядків.

Крайові задачі для звичайних диференціальних рівнянь. Метод прогонки. Метод колокації. Метод найменших квадратів. Метод Гальоркіна. Метод скінченних різниць для розв’язування звичайних диференціальних рівнянь та рівнянь у частинних похідних. Застосування до задач другого порядку: крайової двоточкової, еліптичної, параболічної, гіперболічної.

Список літератури [4 — 5]

3. Навчальна дисципліна «Об’єктно-орієнтоване програмування»

Основні принципи ОО-програмування та схеми реалізації програм на базі простої моделі ОО-мови. ОО-програма представляє собою сценарій взаємодії сукупності об’єктів, що поєднують дані та операції над ними в єдину сутність і є представниками своїх класів, причому останні можуть бути пов’язані в ієрархії наслідування та композиції. Іспит передбачає перевірку знань основних принципів ООП:

Інкапсуляція, наслідування, віртуальні методи та поліморфізм, графічна мова опису класів та їх ієрархії UML.

Також перевіряються знання особливостей ОО-мови C++:

Базовий синтаксис, методи та оператори, їх перевантаження, статичні та константні члени класу, препроцесор, робота з текстовими рядками, потоки введення-виведення, вказівники та посилання, стек та динамічна пам’ять.

Список літератури [6 — 10]

4. Навчальна дисципліна «Операційні системи»

Поняття операційної системи, її призначення. Історія розвитку операційних систем. Види операційних систем. Структура операційної системи. Системні виклики.

Процеси, потоки та методи їх синхронізації. Критичні області, семафори, м’ютекси, події. Класичні проблеми синхронізації та їх вирішення. Планування. Алгоритми планування. Політика та механізм планування. Умови взаємоблокування. Стратегії усунення проблеми.

Системи керування пам’яттю. Підкачка та віртуальна пам’ять. Алгоритми

заміщення сторінок. Моделювання алгоритмів заміщення сторінок. Розробка та реалізація систем зі сторінковою організацією пам’яті.

Принципи роботи апаратури введення-виводу. Переривання. Принципи побудови програмного забезпечення введення-виводу. Обробники переривань. Драйвери пристроїв. Типи пристроїв введення-виводу. Диски, таймери, алфавітно-цифрові термінали, графічні інтерфейси користувача, мережеві термінали.

Файли. Каталоги. Реалізації та приклади файлових систем.

Список літератури [11 — 14]

5. Навчальна дисципліна «Крос-платформне програмування»

[Студенти мають продемонструвати знання підходів до крос-платформного програмування, вміння будувати бізнес-процеси для їх запуску на різних програмно-апаратних платформах, вміння обирати патерни проектування для ефективної розробки бізнес-процесів. Після закінчення курсу студенти вміють та володіють навичками програмування мовою Java, Python або C++ для різних програмних та апаратних платформ. Іспит передбачає перевірку знань основних принципів розробки крос-платформного програмного забезпечення]

Концепція застосування крос-платформного програмування. Цілі та задачі крос-платформного програмування.

Постановка задачі на розробку крос-платформного додатку.

Огляд основних операційних систем і основні методи створення крос-платформного програмного забезпечення.

Основні архітектури процесорів та проблеми сумісності програмного забезпечення (big endian, little endian архітектури).

Особливості процесу портування програмного забезпечення в різних фреймворках графічного інтерфейсу користувача.

Віртуальні машини та крос-платформне програмування.

Регістрові та стекові віртуальні машини. Віртуальна машина LLVM (Low Level Virtual Machine).

Огляд скриптових мов програмування та їх використання у крос-платформному програмуванні.

Типові архітектури крос-платформного програмного забезпечення. Використання MVC, MVP, Command Pattern.

Програмування крос-платформних систем за допомогою скриптових мов програмування.

Засоби мобільної розробки нативного так крос-платформного програмного забезпечення.

Список літератури [15 — 20]

ПРИКІНЦЕВІ ПОЛОЖЕННЯ

Користування допоміжним матеріалом на екзамені
— Дозволяється використання інженерних калькуляторів.

Критерії оцінювання (за системою ECTS, стобальна шкала)

Розв’язання кожної задачі оцінюється за такими критеріями:

95—100	—	задачу розв’язано повністю, вірно
85—94	—	задачу розв’язано вірно, відповідь правильна, але наявними є один-два недоліки (наявними є деякі методичні помилки, порушено послідовність викладок тощо)
75—84	—	задачу розв’язано вірно, але відповідь неправильна (наявними є арифметичні помилки)
65—74	—	задачу розв’язано не повністю, але намічено правильний хід розв’язування
60—64	—	задачу не розв’язано, але наведено формули або твердження, що можуть бути використані при розв’язуванні задачі
менше 60	—	задачу не розв’язано

Результат роботи обчислюється як середнє арифметичне оцінок, що їх отримано за кожну задачу, і заокруглюється до цілих.

Оскільки «Правила прийому до КПІ ім. Ігоря Сікорського в 2023 році» вимагають при обчисленні конкурсного балу застосування шкали оцінювання 100…200 балів, виконується перерахунок оцінки рейтингової системи оцінювання в шкалу єдиного вступного іспиту з іноземної мови відповідно до Таблиці відповідності оцінок.

Таблиця відповідності оцінок РСО (60…100 балів)
оцінкам 200-бальної шкали (100…200 балів)

шкала РСО	шкала 100…200		шкала РСО	шкала 100…200		шкала РСО	шкала 100…200		шкала РСО	шкала 100…200
60	100		70	140		80	160		90	180
61	105		71	142		81	162		91	182
62	110		72	144		82	164		92	184
63	115		73	146		83	166		93	186
64	120		74	148		84	168		94	188
65	125		75	150		85	170		95	190
66	128		76	152		86	172		96	192
67	131		77	154		87	174		97	194
68	134		78	156		88	176		98	196
69	137		79	158		89	178		99	198
									100	200

Приклади типових завдань комплексного фахового випробування

1. Навчальна дисципліна «Дослідження операцій»

Приклад 1. Визначити мінімум функції $f(x) = (40 - 90x)^2$ на інтервалі невизначеності $0 \le x \le 1$ за допомогою методу «золотого перетину». Дозволяється обмежитись трьома кроками зменшення інтервалу.

Розв'язок:

Крок 1. $I_1 = (0,1)$; $L_1 = 1$.

Виконаємо два перших обчислення функції:

$x_1 = \tau = 0.618$; $f(x_1) = 244.0$;

$x_2 = 1 - \tau = 0.382$; $f(x_1) = 31.6$.

Оскільки $f(x_2) < f(x_1)$ та $x_2 < x_1$, інтервал $x > x_1$ виключаємо.

Крок 2. $I_2 = (0,0.618)$; $L_2 = 0.618$.

Наступне обчислення функції виконуємо в точці:

$x_3 = 0.618 \cdot 0.618 = 0.236$; $f(x_3) = 352.0$;

Оскільки $f(x_3) > f(x_2)$ та $x_3 < x_2$, інтервал $x < x_3$ виключаємо.

Крок 3. $I_2 = (0.236,0.618)$; $L_3 = 0.382$.

Наступне обчислення функції виконуємо на відстані $\tau \times L_3$ від лівої границі інтервалу, або, що те саме, на відстані $(1 - \tau) \times L_3$ від його правої границі:

$x_4 = 0.618 - (1 - \tau)L_3 = 0.472$; $f(x_4) = 6.15$;

Оскільки $f(x_4) < f(x_2)$ та $x_4 > x_2$, інтервал $x < x_2$ виключаємо.

В результаті отримаємо наступний інтервал невизначеності: $0.382 < x < 0.618$. Вказане зменшення інтервалу невизначеності проти початкового отримано за 4 оцінки цільової функції.

2. Навчальна дисципліна «Чисельні методи»

Приклад 1. Розв'язати систему лінійних рівнянь методом LU-розкладу:

```
| 2  3  1 ||x(1)| |13|
| 2  4  3 ||x(2)|=|20|
| 2  4  4 ||x(3)| |22|
```

Розв'язок:

Знайдемо LU-розклад матриці системи:

```
U(1,1) = A(1,1) = 2.0000
U(1,2) = A(1,2) = 3.0000
U(1,3) = A(1,3) = 1.0000
L(2,1) = A(2,1)/U(1,1) = 1.0000
L(3,1) = A(3,1)/U(1,1) = 1.0000
U(2,2) = A(2,2) - L(2,1)*U(1,2) = 1.0000
U(2,3) = A(2,3) - L(2,1)*U(1,3) = 2.0000
L(3,2) = (A(3,2)-L(3,1)*U(1,2))/U(2,2) = 1.0000
U(3,3) = A(3,3)-L(3,1)*U(1,3)-L(3,2)*U(2,3) = 1.0000
```

Отже,

```
    | 1  0  0 |           | 2  3  1 |
L = | 1  1  0 |;      U = | 0  1  2 |.
    | 1  1  1 |           | 0  0  1 |
```

Тепер розв'язуємо систему $Ly = b$:

```
        | 1  0  0 ||y(1)| |13|
        | 1  1  0 ||y(2)|=|20|;
        | 1  1  1 ||y(3)| |22|
   y(1) = 13;
   y(2) = 20 - y(1) = 7;
   y(3) = 22 - y(1) - y(2) = 2.
```

Після цього розв'язуємо систему $Ux = y$:

```
        | 2  3  1 ||x(1)| |13|
        | 0  1  2 ||x(2)|=|7|;
        | 0  0  1 ||x(3)| |2|
   x(3) = 2;
   x(2) = 7 - 2*x(3) = 3;
   x(1) = (13 - x(3) - 3*x(2))/2 = 1.
```

Відповідь**:** $x = [1\ 3\ 2]^T$.

Приклад 2. Розв'язати задачу Коші для диференціального рівняння:

$$y' = -2y; \quad y(0) = 1;$$

зробивши 5 кроків розміром $h = 0.1$ методом Ейлера 1 порядку. Оцінити похибку розв'язку, порівнявши знайдене значення в кінцевій точці відрізку інтегрування з аналітичним розв'язком задачі.

Розв'язок: Для $t = 0$ згідно заданої початкової умови маємо $y(0) = 1.0$. Далі продовжуємо обчислення за формулою

$$y_{n+1} = y_n + h \cdot y'(t_n).$$

Знаходимо:

$t_1 = 0.1;$ $\quad y_1 = 1 + 0.1 \cdot (-2 \cdot 1.0) = 0.8;$

$t_2 = 0.2;$ $\quad y_2 = 0.8 + 0.1 \cdot (-2 \cdot 0.8) = 0.64;$

$t_3 = 0.3;$ $\quad y_3 = 0.64 + 0.1 \cdot (-2 \cdot 0.64) = 0.512;$

$t_4 = 0.4;$ $\quad y_4 = 0.512 + 0.1 \cdot (-2 \cdot 0.512) = 0.4096;$

$t_5 = 0.5;$ $\quad y_5 = 0.4096 + 0.1 \cdot (-2 \cdot 0.4096) = 0.32768.$

Аналітичний розв'язок розглядуваної задачі Коші має вигляд:

$y(t) = e^{-2t}$.

Для $t = 0.5$ знаходимо $y(0.5) = 0.3678$.

Похибка інтегрування:

$$\Delta = 0.3678 - 0.32768 = 0.040.$$

А відносна похибка:

$$\varepsilon = 0.040/0.3678 \cdot 100\% \approx 10\%.$$

3. Навчальна дисципліна «Об’єктно-орієнтоване програмування»

Приклад 1. Нижче наведено код, в якому описуються класи для реалізації умовної “програми-дієтолога” (автоматичний підрахунок калорій залежно від кількості вжитих продуктів та способів їх приготування). В коді бракує кількох методів. Ієрархія наслідування класів страв (англ. dish) включає “просту страву” (BasicDish) з одного продукту та “складну (комплексну) страву” (ComplexDish), реалізовану через патерн “компонувальник” (англ. “composite”), до складу якої входять інші страви. До інтерфейсу страви входять операції зі зміни ваги та питомої калорійності на 100 г. продукту, а також операцій отримання ваги та загальної калорійності страви. Реалізація обробки страв виконана на основі патерну “відвідувач” (англ. “visitor”), для чого створено ієрархію “кухонного приладдя” (англ. kitchen tool). Реалізовано два “кухонні прилади”: “сковорідка“ для смаження страв (нехай вона збільшує калорійність страви в півтора рази та зменшує вагу на 20%) та “аналізатор вмісту” для виведення на екран звіту з калорійності страви та її складових. **Завдання**:

1) Намалювати **UML діаграму**, що включає усі класи в коді, за виключенням стандартних (таких, як std::string).
2) Написати **код двох пропущених методів** для зчитування та зміни ваги для класу ComplexDish (вага має змінюватися пропорційно для всіх складових, тобто якщо вага усієї страви збільшилась вдвічі, то це означає, що вага кожної складової має збільшитись удвічі). Інший код за межами даних методів дописувати не можна.
3) Написати, що буде **виведено на екран** в процесі виконання програми.

```
#include <iostream>
#include <string>
#include <list>

// forward declarations of dish types
class BasicDish;
class ComplexDish;

// interfaces
class IKitchenTool {
    public:
        virtual void process(BasicDish* dish) = 0;
        virtual void process(ComplexDish* dish) = 0;
};

class IDish {
    public:
        virtual std::string getDescription() = 0;
        virtual float getWeight() = 0;
        virtual float getCalories() = 0;
        virtual void setWeight(float w) = 0;
        virtual void setCaloriesPerWeight(float cpw) = 0;
        virtual void apply(IKitchenTool* tool) = 0;
};
```

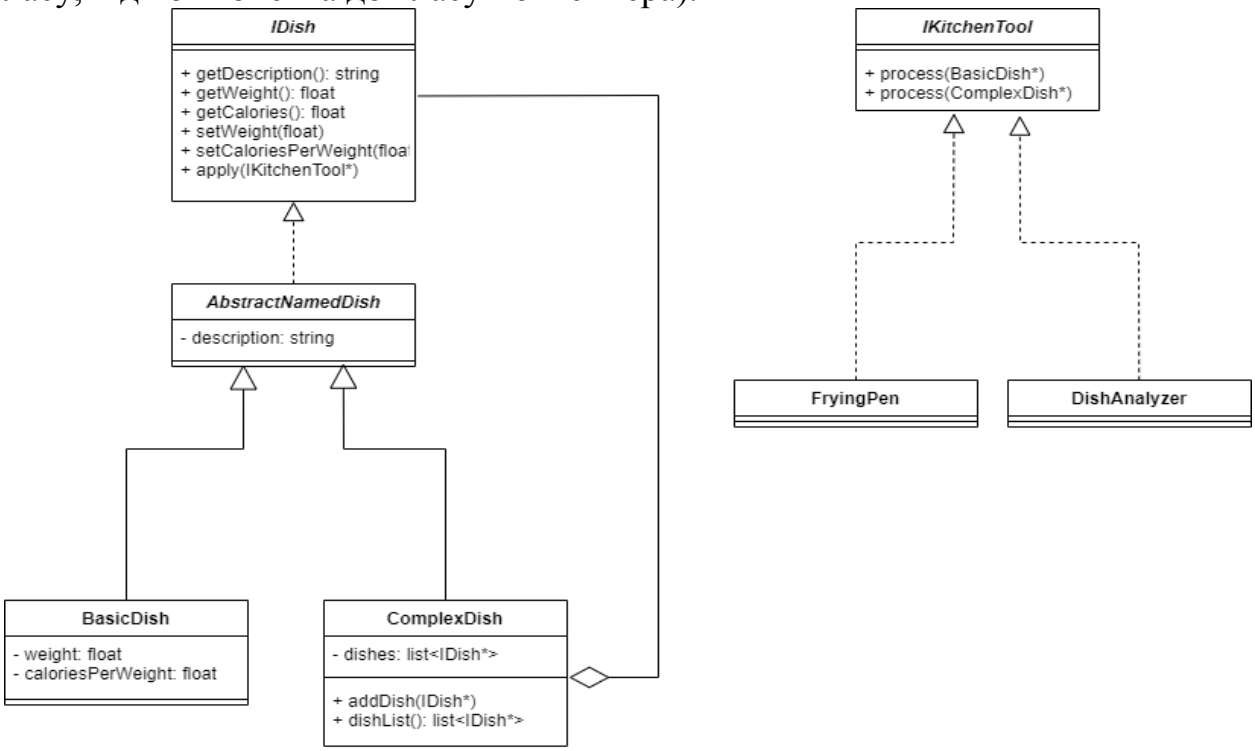
```
// classes
class AbstractNamedDish: public IDish {
    public:
        AbstractNamedDish(std::string desc):
            description(desc) {
        }
        virtual std::string getDescription() override {
            return description;
        }
    private:
        std::string description;
};

class BasicDish: public AbstractNamedDish {
    public:
        BasicDish(std::string desc, float w, float cpw):
            AbstractNamedDish(desc), weight(w), caloriesPerWeight(cpw) {
        }
        virtual float getWeight() override {
            return weight;
        }
        virtual float getCalories() override {
            return caloriesPerWeight * weight / 100;
        }
        virtual void setWeight(float w) override {
            weight = w;
        }
        virtual void setCaloriesPerWeight(float cpw) override {
            caloriesPerWeight = cpw;
        }
        virtual void apply(IKitchenTool* tool) override {
            tool->process(this);
        }
    private:
        float weight;
        float caloriesPerWeight;
};

class ComplexDish: public AbstractNamedDish {
    public:
        ComplexDish(std::string desc):
            AbstractNamedDish(desc) {
        }
        void addDish(IDish* sub_dish) {
            dishes.push_back(sub_dish);
        }
        std::list<IDish*>& dishList() {
            return dishes;
        }
        virtual float getCalories() override {
            float cal = 0;
            for (auto sub_dish: dishes) {
                cal += sub_dish->getCalories();
            }
            return cal;
        }
        virtual void setCaloriesPerWeight(float cpw) override {
```

```
            float cpwKoef = cpw / (getCalories() / getWeight());
            for (auto sub_dish: dishes) {
                sub_dish->setCaloriesPerWeight(sub_dish->getCalories() /
sub_dish->getWeight() * cpwKoef);
            }
        }
        virtual void apply(IKitchenTool* tool) override {
            tool->process(this);
        }
    private:
        std::list<IDish*> dishes;
};

class FryingPan: public IKitchenTool {
    public:
        virtual void process(BasicDish* dish) override {
            dish->setCaloriesPerWeight(1.5 * dish->getCalories() /
(dish->getWeight() / 100));
            dish->setWeight(0.8 * dish->getWeight());
        }
        virtual void process(ComplexDish* dish) override {
            process((BasicDish*) dish);
        }
};

class DishAnalyzer: public IKitchenTool {
    public:
        virtual void process(BasicDish* dish) override {
            std::cout << dish->getDescription()
                << " has total " << dish->getCalories() << " kcal"
                << " in " << dish->getWeight() << " g" << std::endl;
        }
        virtual void process(ComplexDish* dish) override {
            process((BasicDish*) dish);
            std::cout << "including:" << std::endl;
            for (auto sub_dish: dish->dishList()) {
                process((BasicDish*) sub_dish);
            }
        }
};

int main() {
    IDish* meat = new BasicDish("chicken", 200, 200);
    IDish* veg = new BasicDish("potato", 100, 50);
    ComplexDish* dinner = new ComplexDish("fried chicken & potatoes");
    dinner->addDish(meat);
    dinner->addDish(veg);

    IKitchenTool* pan = new FryingPan;
    IKitchenTool* analyzer = new DishAnalyzer;

    meat->apply(analyzer);
    veg->apply(analyzer);
    veg->apply(pan);
    dinner->apply(pan);
    dinner->apply(analyzer);
}
```

Розв’язок.

[Успішне виконання завдання передбачає володіння базовими навичками написання класів та створення об’єктів, знання принципів інкапсуляції, наслідування (успадкування), поліморфізму (реалізованого через віртуальні методи), а також нотації UML (а саме – правил побудови діаграми класів). Бажаним є знання патернів проектування (англ. design patterns). Завдання не спрямоване на оцінку рівня володіння конкретною мовою програмування, і хоча код наведено на мові C++, але з уникненням специфічних лише для цієї мови конструкцій, що дає змогу зрозуміти код тим, хто надає перевагу ОО-мовам Java чи C#. Виключення складають: списки ініціалізації в конструкторі (йдуть між двокрапкою та тілом конструктора, аналогічні звичайним присвоюванням початкових значень полів в тілі конструктора в інших мовах), оголошення чистих віртуальних методів (закінчуються на “= 0”) як частини “інтерфейсних” абстрактних класів, конструкції потокового виведення на екран (std::cout). В коді не використовуються “розумні вказівники” та майже не використовуються посилання, але використовуються звичайні вказівники (позначені через символ “зірочка” та наближені до посилань в інших мовах), при цьому замість оператора “точка” для доступу до членів класу використовується оператор “стрілочка”. Також наявна з часів стандарту C++11 спрощена конструкція циклу по елементам контейнеру (англ. “range-based FOR loop”, з використанням ключового слова auto, аналогічна конструкціям типу “for each” у інших мовах). Код спрощено для збереження лише важливих для виконання завдання елементів, і він не є оптимальним ні з точки зору продуктивності, ані надійності чи стилю, але є робочим. У екзаменаційних завданнях не передбачається використання інших конструкцій мови C++, які не зустрічалися б у даному прикладі.]

1. Почнемо з побудови діаграми наявних класів. Головне тут правильно показати відношення між класами (агрегація, композиція, наслідування - всі вони “однонаправлені” з різними типами стрілок: від наслідника до базового класу, від компонента до класу-контейнера).



2. З коду інтерфейсу IDish видно, що клас ComplexDish має реалізувати методи getWeight() та setWeight(). Для зчитування загальної ваги страви

достатньо просумувати вагу кожної страви-складової, а для задання нової ваги слід вирахувати коефіцієнт зміни ваги та використати його при заданні нової ваги кожної складової. Код може бути, наприклад, таким (не забороняється використовувати іншу, більш довгу форму циклу з оголошенням змінної-ітератора, головне - реалізувати алгоритм пропорційної зміни ваги згідно завдання):

```
virtual float getWeight() override {
    float w = 0;
    for (auto sub_dish: dishes) {
        w += sub_dish->getWeight();
    }
    return w;
}

virtual void setWeight(float w) override {
    float weightKoef = w / getWeight();
    for (auto sub_dish: dishes) {
        sub_dish->setWeight(sub_dish->getWeight() * weightKoef);
    }
}
```

3. Для визначення, що буде виведено на екрані, доцільно почати аналіз з точки входу в програму – функції main(). Хід міркування може бути наступним. Важливо акуратно виконувати обчислення у вірній послідовності.

3.1. У перших двох рядках створено об’єкти базових страв для позначення “курячого м’яса” (meat, вага порції 200 г, питома калорійність на 100 г - 200 ккал) та “картоплі” (veg, 100 г та 50 ккал на 100 г відповідно):

meat.weight = 200 [г]

meat.caloriesPerWeight = 200 [ккал на 100 г]

veg.weight = 100 [г]

veg.caloriesPerWeight = 50 [ккал на 100 г]

3.2. Далі створюється комплексна страва - “вечеря” (dinner), до якої додаються два попередньо створені об’єкти-інгрідієнти meat та veg (точніше - вказівники на них). Далі створюються два об’єкти-”прилади” – сковорідка (pan) для смаження та аналізатор складу страви (analyzer). Вони є реалізацією шаблону “відвідувач”, і далі “відвідують” об’єкти-страви для здійснення операцій над ними.

3.3. Спочатку аналізатор передається об’єктам базових страв через виклик методу apply(), реалізація якого викликає метод process() відвідувача-аналізатора з сигнатурою, що відповідає аргументу типу BasicDish*. Аналізатор виводить на екран назву страви, її калорії та вагу за допомогою викликів відповідних методів класу BasicDish. З урахуванням того, що загальна калорійність є добутком питомої калорійності на 100 г на вагу у грамах, розділену на 100:

*meat.getCalories() = meat.caloriesPerWeight * meat.weight / 100 = 200 * 200 / 100 = 400 [ккал]*

*veg.getCalories() = veg.caloriesPerWeight * veg.weight / 100 = 50 * 100 / 100 = 50 [ккал]*

На екрані буде виведено:

```
chicken has total 400 kcal in 200 g
```

```
potato has total 50 kcal in 100 g
```

3.4. Далі повертаючись до функції main(), у об’єкта veg (овоч - “картопля”) викликається метод apply(), куди передається відвідувач (прилад-”сковорідка” pan). Аналогічно відбувається специфічна для патерну “відвідувач” процедура подвійної диспетчеризації, в результаті якої викликається метод process() класу FryingPan з аргументов BasicDish*. Реалізація цього методу збільшує питому калорійність об’єкта-картоплі у 1.5 рази, а вагу множить на 0.8:

*veg.caloriesPerWeight = 1.5 * veg.getCalories() / (veg.weight / 100) = 1.5 * 50 / (100 / 100) = 75 [ккал на 100 г],*

*veg.weight = 0.8 * veg.weight = 0.8 * 100 = 80 [г].*

3.5. Далі відбувається відвідування об’єктом pan комплексної страви “вечеря” (dinner), куди входить вказівник на щодно змінений об’єкт-”картоплю” (veg) та досі не змінений об’єкт-”м’ясо” (meat). У підсумку відбувається виклик методу process() класу FryingPan з аргументов ComplexDish*. Реалізація цього методу передбачає виклик того самого коду, що і для методу process з аргументом BasicDish*, що реалізовано через оператор приведення типу. Але при розрахунках слід зважати, що будуть поліморфно викликані віртуальні методи об’єкта типу ComplexDish*. Розберемо покроково:

а) розрахуємо аргумент для dish->setCaloriesPerWeight():

*arg(dish->setCaloriesPerWeight()) = 1.5 * dinner.getCalories() / (dinner.getWeight() / 100)*

dinner.getCalories() = meat.getCalories() + veg.getCalories()

*veg.getCalories() = veg.caloriesPerWeight * veg.weight / 100 = 75 * 80 / 100 = 60 [ккал]*

dinner.getCalories() = 400 + 60 = 460 [ккал]

dinner.getWeight() = meat.weight + veg.weight = 200 + 80 = 280 [г]

*arg(dish->setCaloriesPerWeight()) = 1.5 * 460 / (280 / 100) = 246.429 [ккал на 100 г]*

б) при виконанні dish->setCaloriesPerWeight(246.429) питома калорійність складових вечері будуе змінена наступним чином (зросте у 1.5 рази):

cpwKoef = 246.429 / (dinner.getCalories() / dinner.getWeight()) = 246.429 / (460 / 280) = 150

*meat.caloriesPerWeight = meat.getCalories() / meat.weight * cpwKoef = 400 / 200 * 150 = 300 [ккал на 100 г]*

*veg.caloriesPerWeight = veg.getCalories() / veg.weight * cpwKoef = 60 / 80 * 150 = 112.5 [ккал на 100 г]*

в) розрахуємо аргумент для dish->setWeight():

*arg(dish->setWeight()) = 0.8 * dinner.getWeight() = 0.8 * 280 = 224 [г]*

г) при виконанні dish->setWeight(224) вага складових вечері будуе змінена наступним чином (зміниться у 0.8 раз):

weightKoef = 224 / dinner.getWeight() = 224 / 280 = 0.8

*meat.weight = meat.weight * weightKoef = 200 * 0.8 = 160 [г]*

*veg.weight = veg.weight * weightKoef = 80 * 0.8 = 64 [г]*

3.6. І, нарешті, в останньому рядку функції main() відбувається виклик методу apply() об’єкта dinner з аргументом, що вказує на аналізатор. Тобто, буде

викликаний метод process() класу DishAnalyzer з аргументом типу ComplexDish*, що вказує на об’єкт dinner. При цьому спочатку буде викликаний код методу process() класу DishAnalyzer з аргументом типу BasicDish* (через оператор приведення типу).

dinner.getCalories() = meat.getCalories() + veg.getCalories()

*meat.getCalories() = meat.caloriesPerWeight * meat.weight / 100 = 300 * 160 / 100 = 480 [ккал]*

*veg.getCalories() = veg.caloriesPerWeight * veg.weight / 100 = 112.5 * 64 / 100 = 72 [ккал]*

dinner.getCalories() = 480 + 72 = 552 [ккал]

dinner.getWeight() = meat.weight + veg.weight = 160 + 64 = 224 [г]

На екрані буде (аналогічно п.3.3):

```
fried chicken and potatoes has total 552 kcal in 224 g
```

Далі додається рядок “including:” та надається аналогічна інформація по складовим:

```
chicken has total 480 kcal in 160 g
potato has total 72 kcal in 64 g
```

Результат:

Загалом буде виведено:

```
chicken has total 400 kcal in 200 g
potato has total 50 kcal in 100 g
fried chicken & potatoes has total 552 kcal in 224 g
including:
chicken has total 480 kcal in 160 g
potato has total 72 kcal in 64 g
```

4. Навчальна дисципліна «Операційні системи»

Приклад 1. П'ять пакетних задач A, B, C, D, E надходять у систему практично одночасно. Очікується, що час їхнього виконання складе 3, 5, 2, 7 і 1 хв відповідно. Їхні встановлені пріоритети складають 3, 4, 2, 5 і 1, причому 5 – найвищий пріоритет. Визначте середній оборотний час для алгоритмів планування: 1) з урахуванням пріоритетів, 2) "першим прийшов – першим обслугований", 3) "найкоротша задача – перша", зневажаючи часом, що витрачається на переключення між процесами. Передбачається, що в кожен момент часу запущена одна задача, що працює аж до завершення. Усі задачі обмежені тільки можливостями процесора.

Розв’язок:

1) Визначимо оборотний час для кожної задачі: tD = 7; tB = 7+5 = 12; tA = 12+3 = 15; tC = 15+2 = 17; tE = 17+1 = 18 та середній оборотний час: tcp = (7+12+15+17+18)/5 = 13,8 хв.

2) Визначимо оборотний час для кожної задачі: tA = 3; tB = 3+5 = 8; tC = 8+2 = 10; tD = 10+7 = 17; tE = 17+1 = 18 та середній оборотний час: tcp = (3+8+10+17+18)/5 = 11,2 хв.

3) Визначимо оборотний час для кожної задачі: tE = 1; tC = 1+2 = 3; tA = 3+3 = 6; tB = 6+5 = 11; tD = 11+7 = 18 та середній оборотний час: tcp = (1+3+6+11+18)/5 =

7,8 хв.

Приклад 2. Якщо використовується алгоритм заміщення сторінок LRU у системі з чотирма сторінковими блоками і вісьмома сторінками, скільки сторінкових переривань відбудеться для послідовності звертань 0172547103 за умови, що чотири сторінкових блоки спочатку порожні?

Розв’язок:

0	1	7	2	5	4	7	1	0	3
0	1	7	2	5	4	7	1	0	3
	0	1	7	2	5	4	7	1	0
		0	1	7	2	5	4	7	1
			0	1	7	2	5	4	7
				0	1	1	2	5	4
					0	0	0	2	5
									2
п	п	п	п	п	п		п	п	п

Відповідь: 9 переривань.

Приклад 3. Комп'ютер має чотири сторінкових блоки. Час завантаження, час останнього доступу і біти R і M для кожної сторінки наведені нижче (час вказаний у тіках системного годинника):

Сторінка	Завантажена	Останнє звертання	R	M
0	126	280	1	0
1	230	265	0	1
2	140	270	0	0
3	110	285	1	1

Яку сторінку вивантажить алгоритм NRU? Яку сторінку вивантажить алгоритм FIFO? Яку сторінку вивантажить алгоритм LRU? Яку сторінку вивантажить алгоритм "друга спроба"?

Розв’язок:

Алгоритм NRU вивантажує сторінку, що має найнижчий клас, який вираховується за станом бітів звертання та модифікації – тобто сторінку 2, у якої ці два біти очищені.

Алгоритм FIFO вивантажує сторінку, що має найраніший час завантаження, не звертаючи уваги на те, як давно вона використовувалася. Тобто FIFO вивантажить сторінку 3.

Алгоритм LRU вивантажує сторінку, що має найраніший час звертання, не звертаючи уваги на те, як давно вона завантажена. Тобто LRU вивантажить сторінку 1.

Алгоритм “друга спроба” є версією FIFO, що дозволяє уникнути проблеми витіснення сторінок, що використовуються часто. У найстарішої сторінки вивчається біт *R*. Якщо він дорівнює 0, сторінка знаходиться у пам’яті довго і не використовується, тобто, вона може бути замінена новою. Якщо біт *R* дорівнює 1, то він очищується, і сторінка переноситься у кінець списку, а час її

завантаження відновлюється на поточний. У нашому випадку найстарішою є сторінка 3, але її біт *R* дорівнює 1. Для неї буде змінений час завантаження на поточний, та очищений біт звертання. Наступною є сторінка 2, яка має чистий біт звертання, тому вона буде вивантажена.

Відповідь: NRU: 2; FIFO: 3; LRU: 1; "друга спроба": 2.

Приклад 4. До системи надходять чотири процеси з наступними параметрами:

Процес	Тривалість	Час надходження до системи	Пріоритет
A	3	0	1
B	5	1	2
C	2	3	3
D	2	9	4 (найбільший)

Планування відбувається за алгоритмом FIFO. Обчисліть час відгуку та час виконання кожного процесу, а також середній час відгуку та середній час виконання.

Розв’язок:

Даний алгоритм планування працює без витіснення, тобто процеси виконуються у тій послідовності, в якій вони поступають у систему та виконують без переривань. Послідовність виконання задач показана у таблиці, що наведена нижче.

Час відгуку вираховується як час від моменту надходження задачі до моменту її завантаження, який можна визначити за таблицею. Тобто, для процесу A цей час дорівнює 0, для процесу B tB = 3 - 1 = 2.

Час виконання вираховується як період часу від надходження задачі у систему до її повного виконання і може бути також визначений за таблицею. Тобто, для процесу A цей час дорівнює 3, для процесу B tB = 8 - 1 = 7.

Середній час відгуку вираховується як сума часів відгуку всіх процесів, що поділена на кількість процесів. Аналогічно знаходимо середній час виконання.

Відповідь:

0	1	2	3	4	5	6	7	8	9	10	11
A	A	A	B	B	B	B	B	C	C	D	D

Час відгуку: tA = 0, tB = 2, tC = 5, tD = 1, tcp = 8/4 = 2.

Час виконання: tA = 3, tB = 7, tC = 7, tD = 3, tcp = 20/4 = 5.

Приклад 5. Скільки часу займе завантаження з диску програми розміром 64 Кбайт, якщо його середній час пошуку дорівнює 5 мс, час обертання – 10 мс, кожна доріжка містить 32 Кбайт, а розмір сторінки дорівнює 8 Кбайт? Сторінки на диску розташовані випадковим образом, і кількість циліндрів є настільки великою, що можна ігнорувати варіант, при якому дві сторінки опиняться на тому самому циліндрі.

Розв’язок:

32 Кбайт зчитуються за 10 мс, тобто час зчитування однієї сторінки розміром 8 Кбайт складає 8*10/32 = 2,5 мс.

Загальний час завантаження підраховується як T = N сторінок * (T пошуку + T зчитування).

Тоді T = 64 Кбайт /8 Кбайт * (5 мс + 2,5 мс) = 8 * 7,5 = 60 мс.

Відповідь: 60 мс.

Приклад 6. У гнучкого диску 40 циліндрів. Операція пошуку займає 6 мс на циліндр. Скільки знадобиться часу для зчитування 50-блокового файлу, якщо затримка обертання складає 50 мс, час переносу одного блоку дорівнює 25 мс, а два логічно послідовних блоки (тобто наступних один за іншим у файлі) виявляться в середньому на відстані 10 циліндрів?

Розв’язок:

Якщо відстань між двома блоками дорівнює 10 циліндрам, то час зчитування 50 блоків дорівнюватиме: T = N блоків * (T затримки + T переносу + T пошуку * N циліндрів) = 50*(50+25+6*10) = 6750 мс = 6,75 с.

Відповідь: 6,75 с.

5. Навчальна дисципліна «Крос-платформне програмування»

Приклад 1. Дано десяткове число ***5***.

Для *unsigned integer* довжиною 4 байти записати дане число у бінарній формі (за допомогою 0 та 1) у форматі **big endian**.

Розв’язок:

Оскільки представлення числа має бути у форматі big endian, то 0x05 0x00 0x00 0x00. А у бінарному форматі буде мати вигляд

Старший байт **Молодший байт**

0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Приклад 2. Дано десяткове число ***3***.

Для *unsigned integer* довжиною 4 байти записати дане число у бінарній формі (за допомогою 0 та 1) у форматі **little endian**.

Розв’язок:

Оскільки представлення числа має бути у форматі little endian, то 0x00 0x00 0x00 0x03. А у бінарному форматі буде мати вигляд

Старший байт **Молодший байт**

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1

СПИСОК ЛІТЕРАТУРИ

1. Зайченко О.Ю. Дослідження операцій Зайченко О.Ю., Зайченко Ю.П. — К: Видавничій дім «Слово», 2014. — 472 с.
2. Навчально-методичний посібник до практичних занять з курсу «Математичні методи оптимізації» для студентів магістратури усіх спеціальностей / Уклад. О.Ю.Зайченко. — К.: Політехніка, 2007. — 88 с.
3. Ладогубець В.В. Алгоритми параметричної оптимізації складних систем / Ладогубець В.В., Ладогубець Т.С., Ладогубець О.В. — К.: АВЕРС, 2006. — 139 с.
4. Фельдман Л.П. Чисельні методи в інформатиці /Петренко А.І., Дмитрієва О.А. — К.: Видавнича група BHV, 2006. — 480 с.
5. Задачин В.М. Чисельні методи: навч. пос. /В.М.Задачин, І.Г.Конюшенко. — Х.: ХНЕУ ім. С. Кузнеця, 2014. — 180 с.
6. Васильєв О. Програмування на С++ в прикладах і задачах. – К.: “Ліра – К”. – 2017. – 382 с. – ISBN 9786177507412.
7. Stroustrup B. A Tour of C++ (C++ In Depth Series, 3rd ed.). – Addison-Wesley Professional. – 2022. – 320 p. – ISBN 9780136816485.
8. Фрімен Е., Робсон Е., Бейтс Б., Сієрра К. Head First. Патерни проєктування. – Х.:“Фабула”. – 2020. – 672 с. – ISBN 9786170961594
9. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. – Addison-Wesley. – 1995. – 395 p. – ISBN 0201633612
10. Booch G. Object-Oriented Analysis and Design with Applications. – Addison-Wesley Professional. – 2007. – 720 p. – ISBN 9780201895513
11. Шеховцов В. А. Операційні системи : підруч. для студ. / В. А. Шеховцов. — К. : Вид. група BHV, 2008. — 576 с.
12. Tanenbaum A. Modern Operating Systems, 4th ed. / Tanenbaum A., Bos H. — Pearson, 2014. — 1136 p.
13. Операційні системи: [Електронний ресурс]: навч. посіб. для студ. спеціальності 123 «Комп’ютерна інженерія» / В. Г. Зайцев, І. П. Дробязко; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 3 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2019. – 240 с.
14. Бондаренко М. Ф. Операційні системи : навч. посібник для вищих навч. закладів за напрямом "Комп'ютерні науки"/ М. Ф. Бондаренко, О. Г. Качко. — Х.: Компанія СМІТ, 2008. — 432 с.
15. Arnold K., Gosling J., Holmes D. The Java Programming Language (4th Edition). Addison-Wesley Professional, 2005. — 928 p.
16. Blanchette J., Summerfield M. C++ GUI Programming with Qt 4 (2nd Edition). Prentice Hall, 2008. — 752 p.
17. Васильєв О. Програмування мовою Java.- Видавництво Навчальна книга Богдан, 2020. - 696 с. ISBN: 9789661058797
18. Мартін Р. Чистий код. Створення і рефакторинг за допомогою Agile. - Харків: Видавництво Фабула, 2019. - 448 с. ISBN 978-617-09-5285-1

19. Мартін Р. Чиста архітектура. Мистецтво створення програмного забезпечення. Видання друге. - Харків: Видавництво Фабула, 2019. - 368 с. ISBN 978-617-09-5286-8
20. Exploring Cross-Platform Development with Flutter, React Native, and Xamarin by Eric Windmill. January 2020. Publisher(s): Manning Publications. - 98 pages. ISBN 9781617296789

Розробники програми:

Булах Б.В., к.т.н., доцент кафедри системного проектування

Богдан БУЛАХ

Безносик О.Ю., к.т.н., доцент кафедри системного проектування

Олександр БЕЗНОСИК

Харченко К.В., к.т.н., доцент кафедри системного проектування

Костянтин ХАРЧЕНКО